Total Recall:
Persistence of Passwords in Android

Jaeho Lee, Ang Chen, Dan S. Wallach

Motivation

Memory is not a safe place for sensitive data.

Unprivileged attackers can access sensitive data from device memory.
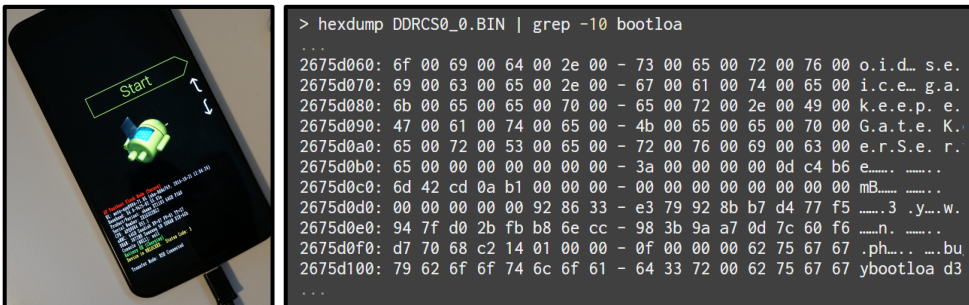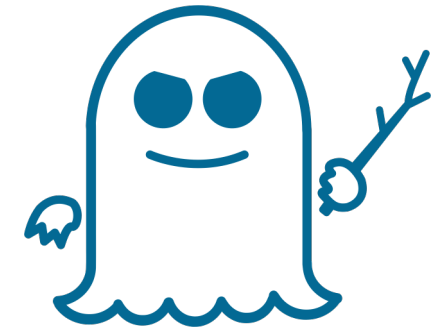
Cold-boot attack

Heartbleed (CVE-2014-0160)

Nexus 5X bootloader vulnerability (ALEPH-2016000)

Meltdown (CVE-2017-5754)

Spectre (CVE-2017-5753)

Memory is not a safe place for sensitive data.

Sensitive data should be deleted as soon as it is no longer in use.

- Crypto libraries have long recognized the importance of this practice.
  - OpenSSL is well engineered to follow the practice.

```
void *OPENSSL_clear_realloc(void *p, size_t old_len, size_t num)
void OPENSSL_clear_free(void *str, size_t num)
void OPENSSL_cleanse(void *ptr, size_t len);
void *CRYPTO_clear_realloc(void *p, size_t old_len, size_t num, const char *file, int line)
void CRYPTO_clear_free(void *str, size_t num, const char *, int)
```

```
void *SSL_SESSION_free(SSL_SESSION *ss) {
    ...
    OPENSSL_cleanse(ss->master_key, sizeof(ss->master_key));
    OPENSSL_cleanse(ss->session_id, sizeof(ss->session_id));
    ...
}
```

RICE

Memory is not a safe place for sensitive data.

Sensitive data should be deleted a

- Crypto libraries have long recogn
  - OpenSSL is well engineered to follo

Removing Secrets from Android's TLS

[NDSS 18]

Jaeho Lee and Dan S. Wallach
Rice University
{jaeho.lee, dwallach} @rice.edu

```c
void *OPENSSL_clear_realloc(void *p, size_t old_len, size_t num)
void OPENSSL_clear_free(void *str, size_t num)
void OPENSSL_cleanse(void *ptr, size_t len);
void *CRYPTO_clear_realloc(void *p, size_t old_len, size_t num, const char *file, int line)
void CRYPTO_clear_free(void *str, size_t num, const char *, int)
```

```c
void *SSL_SESSION_free(SSL_SESSION *ss) {
    ...
    OPENSSL_cleanse(ss->master_key, sizeof(ss->master_key));
    OPENSSL_cleanse(ss->session_id, sizeof(ss->session_id));
    ...
}
```

RICE

Focus of Research

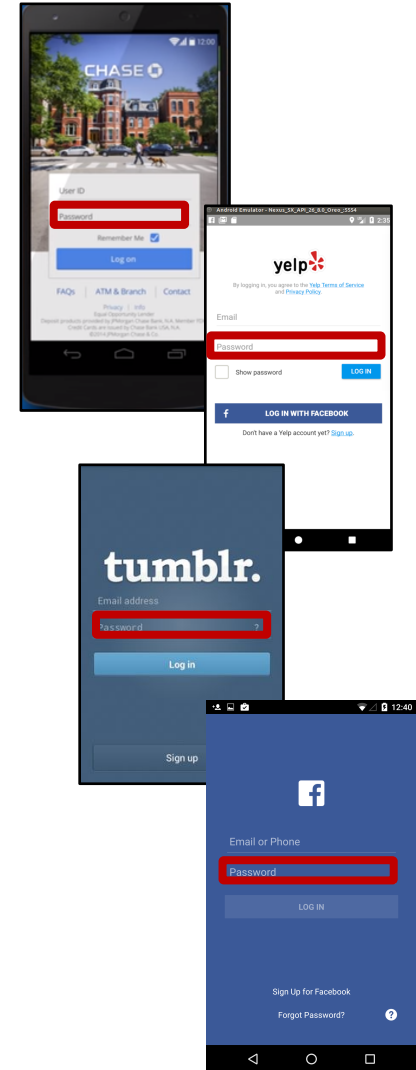**What about user input passwords in Android?**

- **User input passwords** are of paramount importance.
  - ➤ Stolen passwords cause widespread damage.
- Numerous 3rd party apps in **Android** require our passwords.

Do apps manage user input passwords well?

Does Android support enough protection for them?

Are they safe under *memory disclosure attacks*?

RICE

Preliminary Study
Is password exposure a real problem in Android?

Preliminary Study
Is password exposure a real problem in Android?

Application	Type	Installs	Passwords
Gmail	Email	1,000 M	6
Chrome	Browser	1,000 M	4
Facebook	Social	1,000 M	6
Tumblr	Social	100 M	4
Yelp	Social	10 M	3
Chase Bank	Finance	10 M	5
1Password	Password	1 M	4
Dashlane	Password	1 M	2
Keepass2Android	password	1 M	1
passwdSafe	password	0.1 M	12
Unlocking process	system	Built-in	7
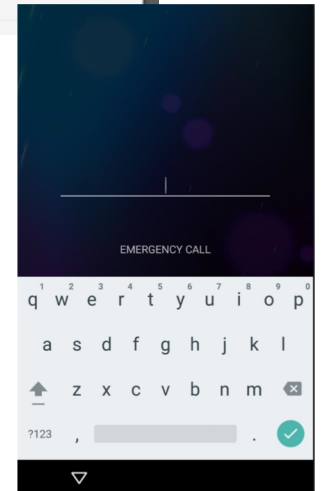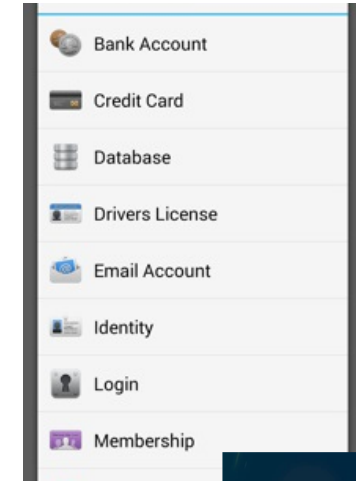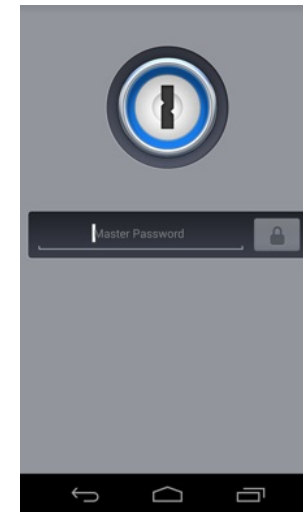
Master password

lockscreen password

RICE

Passwords are vulnerable to memory disclosure attack.

Password strings are easily recognizable from the binary dump.

Facebook

```
64 69 64 3D 26 66 6F 72 6D 61 74 | nf_fbm.=NO_FILE&adid=&format
31 62 32 33 39 2D 31 32 33 34 2D | =json&device_.dbc1b239-1234-
34 31 32 33 34 26 65 6D 61 69 6C | 4567-9108-6f0e12341234&email
70 61 73 73 77 6F 72 64 3D 52 65 | =123456789012345&password=Re
6C 3D 74 72 75 65 26 66 61 6D 69 | ASCII PASSWORD &cpl=true&fami
79 70 65 3D 04 62 61 73 65 64 5F | ly_,credentials_type=.based_
```

Tumblr

```
45d56b5e 00 00 78 00 5F 00 61 00 75 00 74 00 68 00 5F 00 75 00 73 00 65 00 72 00 6E 00 61 00 6D ..x._.a.u.t.h._.u.s.e.r.n.a.m
45d56b7b 00 65 00 3D 00 52 00 65 00 61 00 6C 00 55 00 73 00 65 00 72 00 6E 00 61 00 6D 00 65 00 .e.=.R.e.a.l.U.s.e.r.n.a.m.e.
45d56b98 68 00 65 00 72 00 65 00 68 00 65 00 72 00 65 00 25 00 34 00 30 00 67 00 6D 00 61 00 69 h.e.r.e.h.e.r.e.%.4.0.g.m.a.i
45d56bb5 00 6C 00 2E 00 63 00 6F 00 6D 00 26 00 78 00 5F 00 61 00 75 00 74 00 68 00 5F 00 70 00 .l...c.o.m.&.x._.a.u.t.h._.p.
45d56bd2 61 00 73 00 73 00 77 00 6F 00 72 00 64 00 3D 00 UTF-16 PASSWORD 6C 00 4D 00 79 00 70 a.s.s.w.o.r.d.=.R.e.a.l.M.y.p
45d56bef 00 61 00 73 00 73 00 77 00 6F 00 72 00 64 00 48 00 65 00 72 00 26 00 78 00 5F 00 61 00 .a.s.s.w.o.r.d.H.e.r.&.x._.a.
```

Attackers only need one memory disclosure vulnerability.

This is an old issue, but still problematic.

  ➤ 5/14 apps hoard passwords in memory (CleanOS [OSDI 12])

RICE

Goal
Answers the two research questions

What causes password retention when passwords are no longer used?
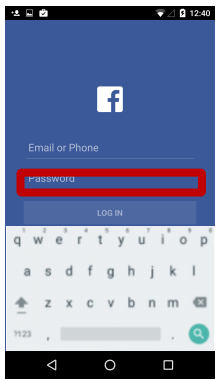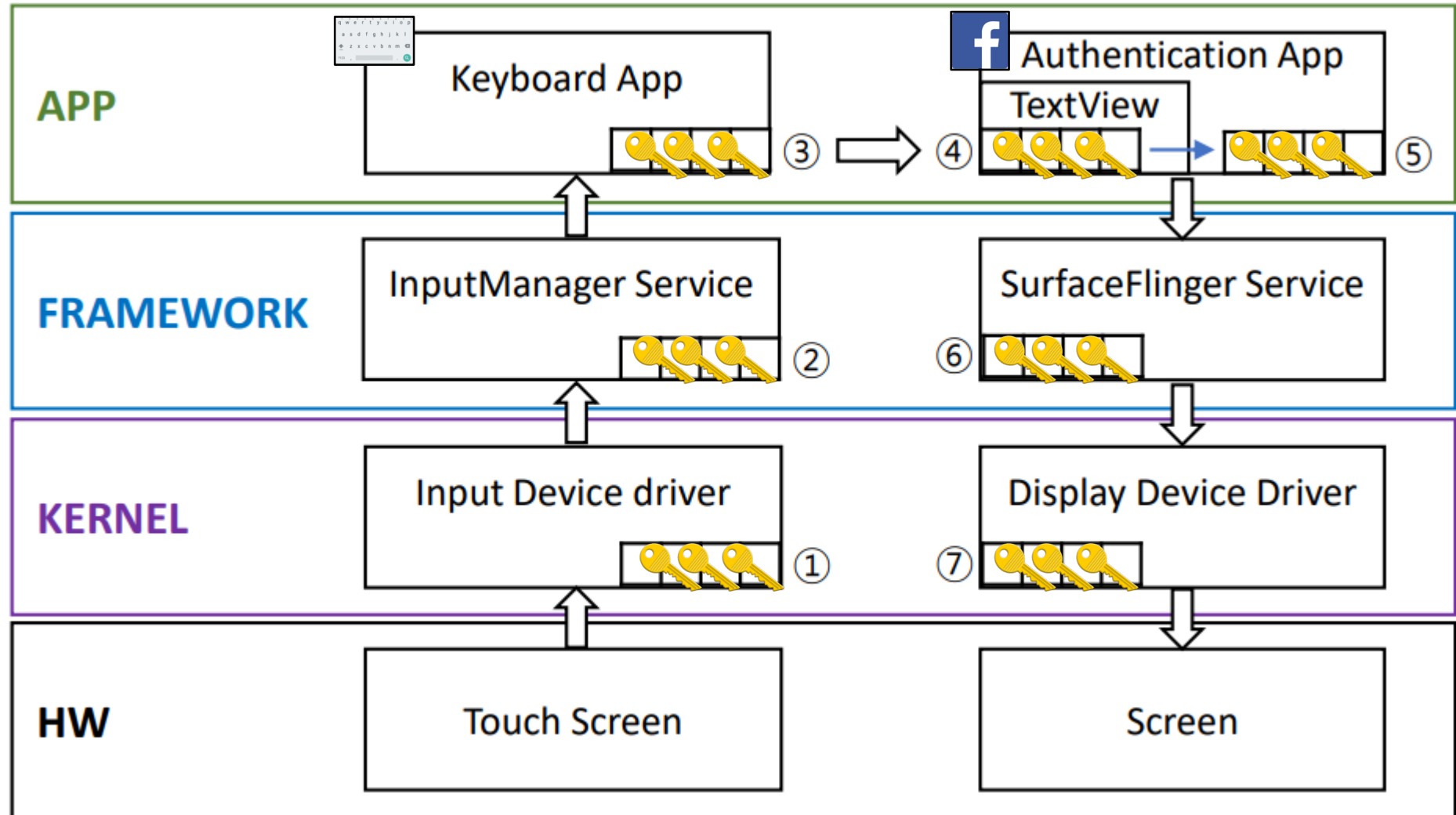
*We analyzed Android framework and various apps,
and found root causes.*

Can we solve the password retention problem effectively?

*Practical solution is possible to reduce passwords,
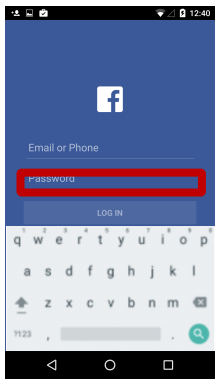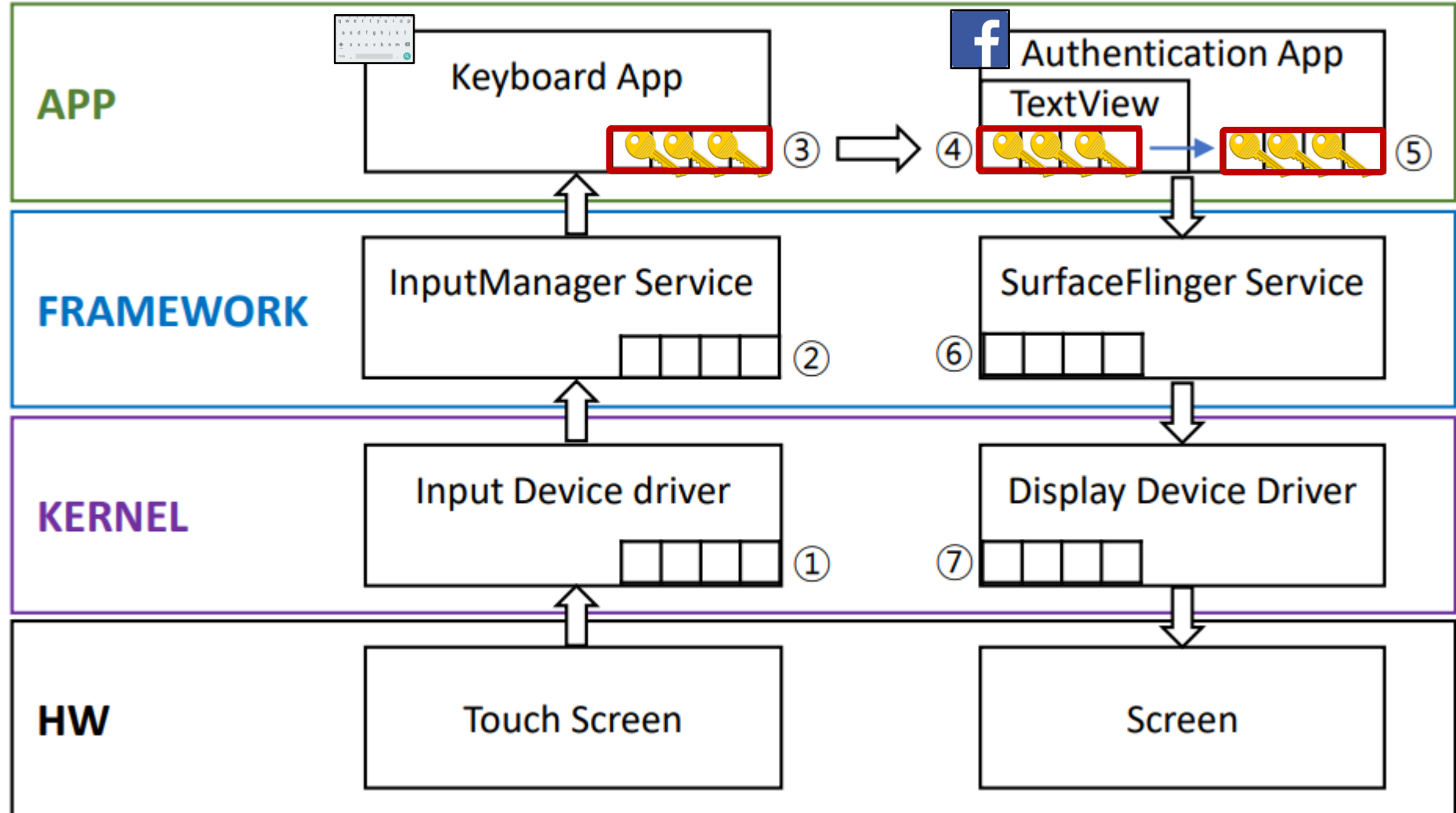with minor change and performance impact.*

RICE

Root causes of password retention
Password flow in Android

Root causes of password retention
Three components retain user passwords unnecessary.

Root causes of password retention
1. Android Framework

Root causes of password retention
Android framework: Keyboard (IME) applications

Default keyboard app buffers recent input regardless of passwords.

Application	Installs	Passwords
LatinIME	Built-in	**2**
Gboard	1,000 M	0
SwiftKey	300 M	0
Go	100 M	**1**
KiKA	100 M	**1**
TouchPal	100 M	**4**
Cheetah	50 M	**7**
FaceMoji	10 M	**1**
New Keyboard	10 M	**1**
Simeji	10 M	0
Simplified Chinese	0.1 M	**135**
Baidu Voice	0.1 M	**2**
TS Korean	0.01 M	0

We dumped the memory of keyboard app process after login.

The insecure default open source may influence 3rd keyboard apps.

We investigated popular keyboard apps.

Captured passwords from 9/13 keyboard apps

Lack of user input password protection in UI implementation.

- No dedicated class for the password widget.

12,000 LoC of `TextView` is reused both for normal and password entry.

- Missing necessary secure handling for passwords.

  - ➤ E.g., the widget holds passwords even though the UI is going to the background.

RICE

Poor API design in TextView

```
public CharSequence getText()

Return the text that TextView is displaying.
```

➢ Developers are guided to store passwords in `String` objects.

`String` is unsuitable for storing sensitive data [Java Crypto Arch. Reference Guide]

➢ `String` objects are immutable.

➢ No method is defined to overwrite the contents.

➢ Always collect and store security sensitive information in a char array.

RICE

Poor API design in TextView

```
public CharSequence getText()

Return the text that TextView is displaying.
The content of the return value should not be modified. Make your own copy first.
```

➤ Developers are guided to store passwords in `String` objects.

Comparing with password widget in Desktop JDK: *JPasswordField*

➤ Corresponding `String getText()` has been deprecated since Java 1.2 ('98).

```
public char[] getPassword()

Returned char[] should be cleared after use by setting each character to zero.
```

RICE

Root causes of password retention
2. Android applications

Developers often implement authentication routines from scratch.

➤ They have different levels of awareness and experience in security.

Various bad practices throughout Android developers.

➤ Sending raw passwords into files or through network.

➤ Widespread use of `String` passwords

– Surprisingly, all apps use `String` passwords except one password manager.

➤ No cleanup passwords after authentication.

RICE

Solution

Solution

The identified causes should be addressed altogether.

Lack of password protection in the Android framework.

SecureTextView

Developers' mistake in managing passwords.

- Encourage the best practice.
  - ➤ Use char array passwords.
  - ➤ Clear the buffer of `TextView` after login.
  - ➤ Derive a strong key and use it instead of raw passwords.
  - ➤ Overwrite all passwords after login.

Abstraction for the best practices
KeyExporter

RICE

Solution: KeyExporter

Developers make mistakes in dealing with passwords.

- Even critical apps

The purpose of using input passwords is the same throughout the apps.
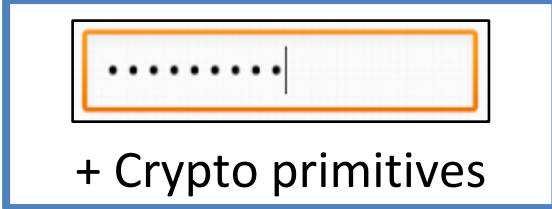
- Developers repeat similar logic.

Make easy for developers to do the right thing.

RICE

Solution: KeyExporter

onClick()

pw = UI.getText().toString()

Acessing passwords

propagate(pw)

k = deriveKey(pw)

verify_user(k)   to_file(k)

decryptDB(k)   to_network(k)

---

password = UI.getText().toString();

Gives what developers actually need.

•••••••••

+ Crypto primitives

---

onClick()

ke = KeyExporter(UI)

k = ke.deriveKey()

verify_user(k)   to_file(k)

decryptDB(k)   to_network(k)
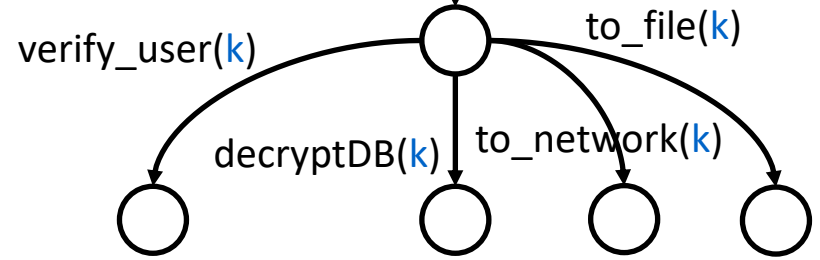
Developers don't need to access pw.

RICE
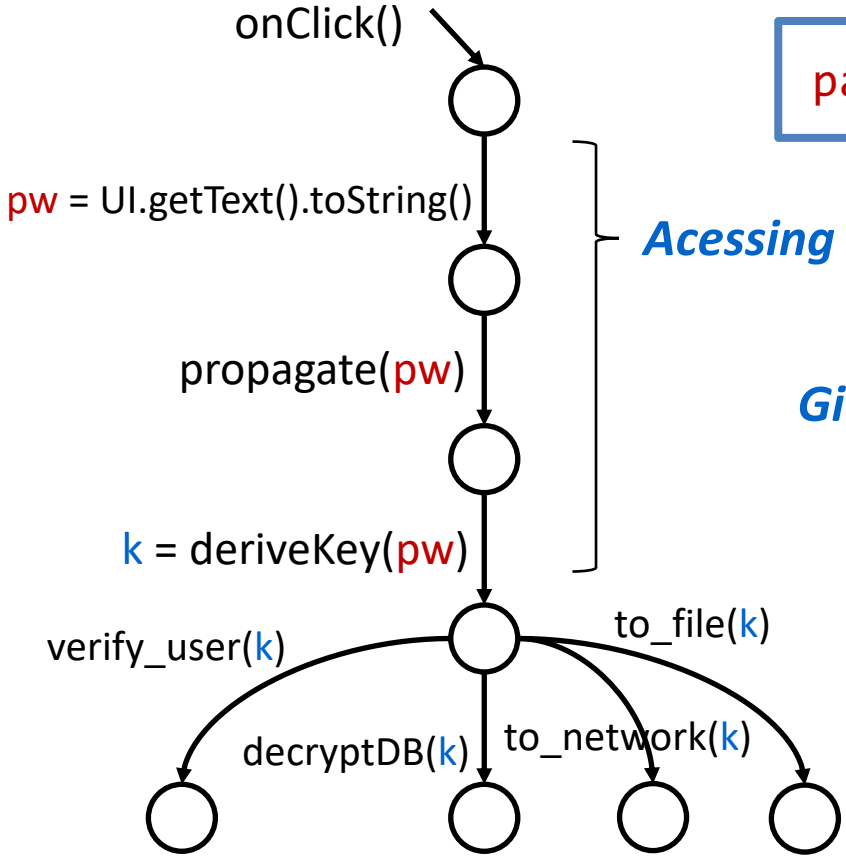
Implementation

Android Framework (patch submitted to Google)

- **SecureTextView**: extension of TextView

- Fix lockscreen processes and LatinIME Keyboard app

- Built on Android 8.1.0_r20

KeyExporter API (unmodified Android)

- Support key derivation functions: PBKDF2, HMAC, Scrypt

- Support PAKE (password-authenticated key agreement): SRP Protocol

RICE

Evaluation

How effective can our solution fix password retention?

Is KeyExporter generally applicable to different types of apps?

Evaluation

Evaluation after integrating KeyExporter with various apps.

Application	Description	Original Android	SecureTextView + KeyExporter
Naïve sample	Sending the raw password to the server	25	0
Secure sample	HMAC-based challenge-response protocol	21	0
Unlocking process	Hash with scrypt and send it to TEE	7	0
passwdSafe	Open source password manager with 40,000 LoC	12	0

Evaluation for close source

Application	Description	Original Android	SecureTextView Only
Yelp	Close source. Log in with Facebook OAuth	3	2

Found in memory of Facebook

Conclusion

Analyzed the Android framework and a variety of apps comprehensively.

- Identified the root causes of password retention.

Developed practical solution without intrusive modification in Android.

- SecureTextView (Android 8.1 framework patch)

- KeyExporter (Standalone libraries)

Evaluated with apps in various categories including popular security app.

RICE

Questions?

Jaeho Lee

jaeho.lee@rice.edu

Source: https://github.com/friendlyjlee/totalrecall

RICE

Back-up Slides

Feedback from Disclosure reporting

Google Android Security Team: Won't fix

In Android, each process runs in its own security sandbox.
Because of this isolation, we don't believe there is a significant advantage in attempting to wipe memory.
Being able to read the memory of another process indicates a system is already significantly compromised.

Facebook Security Team: No immediate plan

The problems are best dealt with by the underlying platform, rather than individually.
If Google chooses to adopt some of your suggestions, we will evaluate them for potential future adoption.

KiKa Keyboard app: Working on it

Kika Keyboard used AOSP LatinIME. So it might be common issue for every IME powered by AOSP LatinIME.
It's not very easy to root, but there are still many users having root Android phone. Thus, it should be fixed AOSP. We are still working on it.

RICE

Related Work

Protecting sensitive data in secure storages.

- TRESOR (Security 11): CPU registers
- CleanOS (OSDI 12): Encrypting data in memory
- Sentry (ASPLOS 15): Cache + iRAM in SoC chip
- CaSE (Oakland 16): Cache + TrustZone
- Ginseng (NDSS 19): CPU registers

General and Backward-compatible

Intrusive modification and significant overhead

Dynamic Analysis: Taint Analysis

- TaintDroid (OSDI 10): Detecting leakage of sensitive data
- K-Hunt (CCS17): Identifying insecure keys.

RICE